





Relaxed Locally Correctable Codes with Improved Parameters

Vahid R. Asadi  

Simon Fraser University, Burnaby, Canada

Igor Shinkar  

Simon Fraser University, Burnaby, Canada

Abstract

Locally decodable codes (LDCs) are error-correcting codes $C: \Sigma^k \rightarrow \Sigma^n$ that admit a local decoding algorithm that recovers each individual bit of the message by querying only a few bits from a noisy codeword. An important question in this line of research is to understand the optimal trade-off between the query complexity of LDCs and their block length. Despite importance of these objects, the best known constructions of constant query LDCs have super-polynomial length, and there is a significant gap between the best constructions and the known lower bounds in terms of the block length.

For many applications it suffices to consider the weaker notion of *relaxed LDCs* (RLDCs), which allows the local decoding algorithm to abort if by querying a few bits it detects that the input is not a codeword. This relaxation turned out to allow decoding algorithms with constant query complexity for codes with *almost linear* length. Specifically, [2] constructed a q -query RLDC that encodes a message of length k using a codeword of block length $n = O_q(k^{1+O(1/\sqrt{q})})$ for any sufficiently large q , where $O_q(\cdot)$ hides some constant that depends only on q .

In this work we improve the parameters of [2] by constructing a q -query RLDC that encodes a message of length k using a codeword of block length $O_q(k^{1+O(1/q)})$ for any sufficiently large q . This construction matches (up to a multiplicative constant factor) the lower bounds of [14, 23] for constant query LDCs, thus making progress toward understanding the gap between LDCs and RLDCs in the constant query regime.

In fact, our construction extends to the stronger notion of relaxed locally *correctable* codes (RLCCs), introduced in [13], where given a noisy codeword the correcting algorithm either recovers each individual bit of the codeword by only reading a small part of the input, or aborts if the input is detected to be corrupt.

2012 ACM Subject Classification Theory of computation \rightarrow Error-correcting codes

Keywords and phrases Algorithmic coding theory, consistency test using random walk, Reed-Muller code, relaxed locally decodable codes, relaxed locally correctable codes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.18

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: <https://eccc.weizmann.ac.il/report/2020/142/> [1]

Acknowledgements We are thankful to Tom Gur for many fruitful discussions on this topic, and we are grateful to the anonymous referees for their valuable suggestions that helped improve the presentation of the paper.

1 Introduction

Locally decodable codes (LDCs) are error-correcting codes that admit a decoding algorithm that recovers each specific symbol of the message by reading a small number of locations in a possibly corrupted codeword. More precisely, a locally decodable code $C: \mathbb{F}^k \rightarrow \mathbb{F}^n$ with local decoding radius $\tau \in [0, 1]$ is an error-correcting code that admits a local decoding



© Vahid R. Asadi and Igor Shinkar;

licensed under Creative Commons License CC-BY 4.0

48th International Colloquium on Automata, Languages, and Programming (ICALP 2021).

Editors: Nikhil Bansal, Emanuela Merelli, and James Worrell; Article No. 18; pp. 18:1–18:12

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



algorithm \mathcal{D}_C , such that given an index $i \in [k]$ and a corrupted word $w \in \mathbb{F}^n$ which is τ -close to an encoding of some message $C(M)$, reads a small number of symbols from w , and outputs M_i with high probability. Similarly, we have the notion of *locally correctable codes (LCCs)*, which are error-correcting codes that not only admit a local algorithm that decodes each symbol of the message, but are also required to correct an arbitrary symbol from the entire codeword. Indeed, note that for systematic codes the notion of LCC is strengthening of LDC. Locally decodable and locally correctable codes have many applications in different areas of theoretical computer science, such as complexity theory, coding theory, property testing, cryptography, and construction of probabilistically checkable proof systems. For details, see the surveys [26, 18] and the references within.

Despite the importance of LDCs and LCCs, and the extensive amount of research studying these objects, the best known construction of constant query LDCs has super-polynomial length $n = \exp(\exp(\log^{\Omega(1)}(k)))$, which is achieved by the highly non-trivial constructions of [25] and [9]. For constant query LCCs, the best known constructions are of exponential length, which can be achieved by some parameterization of Reed-Muller codes. It is important to note that there is a huge gap between the best known lower bounds for the length of constant query LDCs and the length of best known constructions. Currently, the best known lower bound on the length of LDCs says that for $q \geq 3$ it must be at least $k^{1+\Omega(1/q)}$, where q stands for the query complexity of the local decoder. See [14, 17, 23] for the best general lower bounds for constant query LDCs.

Motivated by applications to probabilistically checkable proofs (PCPs), Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan introduced in [2] the notion of *relaxed locally decodable codes (RLDCs)*. Informally speaking, a relaxed locally decodable code is an error-correcting code which allows the local decoding algorithm to abort if the input codeword is corrupt, but does not allow it to err with high probability. In particular, the decoding algorithm should always output the correct symbol, if the given word is not corrupted. Formally, a code $C: \mathbb{F}^k \rightarrow \mathbb{F}^n$ is an RLDC with decoding radius $\tau \in [0, 1]$ if it admits a relaxed local decoding algorithm \mathcal{D}_C which given an index $i \in [k]$ and a possibly corrupted codeword $w \in \mathbb{F}^n$, makes a small number of queries to w , and satisfies the following properties.

Completeness: If $w = C(M)$ for some $M \in \mathbb{F}^k$, then $\mathcal{D}_C^w(i)$ should output M_i .

Relaxed decoding: If w is τ -close to some codeword $C(M)$, then $\mathcal{D}_C^w(i)$ should output either M_i or a special *abort* symbol with probability at least $2/3$.

This relaxation turns out to be very helpful in terms of constructing RLDCs with better block length. Indeed, [2] constructed a q -query RLDC with block length $n = k^{1+O(1/\sqrt{q})}$.

The notion of *relaxed LCCs (RLCCs)*, recently introduced in [13], naturally extends the notion of RLDCs. These are error-correcting codes that admit a correcting algorithm that is required to correct every symbol of the codeword, but is allowed to abort if noticing that the given word is corrupt. More formally, the local correcting algorithm gets an index $i \in [n]$, and a (possibly corrupted) word $w \in \mathbb{F}^n$, makes a small number of queries to w , and satisfies the following properties.

Completeness: If $w \in C$, then $\mathcal{D}_C^w(i)$ should output w_i .

Relaxed correcting: If w is τ -close to some codeword $c^* \in C$, then $\mathcal{D}_C^w(i)$ should output either c_i^* or a special *abort* symbol with probability at least $2/3$.

Note that if the code C is systematic, i.e., the encoding of any message $M \in \mathbb{F}^k$ contains M in its first k symbols, then the notion of RLCC is stronger than RLDC.

Recently, building on the ideas from [13], [3] constructed RLCCs whose block length matches the RLDC construction of [2]. For the lower bounds, the only result we are aware of is the work of Gur and Lachish [12], who proved that for any RLDC the block length must be at least $n = k^{1+\Omega(1/q^2)}$.

Given the gap between the best constructions and the known lower bounds, it is natural to ask the following question:

What is the best possible trade-off between the query complexity and the block length of an RLDC and RLCC?

In particular, [2] asked whether it is possible to obtain a q -query RLDC whose block length is strictly smaller than the best known lower bound on the length of LDCs. A positive answer to their question would show a separation between the two notions, thus proving that the relaxation is *strict*. See paragraph *Open Problem* in the end of Section 4.2 of [2].

In this work we make progress on this problem by constructing a relaxed locally decodable code $C: \mathbb{F}^K \rightarrow \mathbb{F}^N$ with query complexity $O(q)$ and block length $K^{1+O(1/q)}$. In fact, our construction gives the stronger notion of a relaxed locally correctable code.

► **Theorem 1 (Main Theorem).** *For every sufficiently large $q \in \mathbb{N}$ there exists an q -query relaxed locally correctable code $C: \{0, 1\}^K \rightarrow \{0, 1\}^N$ with constant relative distance and constant decoding radius, such that the block length of C is*

$$N = q^{O(q^2)} \cdot K^{1+O(1/q)} .$$

Furthermore, the code C is linear and systematic.

Therefore, our construction improves the parameters of the q -query RLDC construction of [2] with block length $N = K^{1+O(\sqrt{1/q})}$, and matches (up to a multiplicative factor in q) the lower bound of $\Omega(K^{1+\frac{1}{\lceil q/2 \rceil - 1}})$ for the block length of q -query LDCs [14, 23].

► **Remark 2.** In this paper we prove Theorem 1 for a code $C: \mathbb{F}^K \rightarrow \mathbb{F}^N$ over a large alphabet. Specifically, we show a code $C: \mathbb{F}^K \rightarrow \mathbb{F}^N$ satisfying Theorem 1, for a finite field \mathbb{F} satisfying $|\mathbb{F}| \geq c_q \cdot K^{1/q}$, for some $c_q \in \mathbb{N}$ that depends only on q .

Using the techniques from [3] it is not difficult to obtain an RLCC over the binary alphabet with almost the same block length. Indeed, this can be done by concatenating our code over large alphabet with an arbitrary binary code with constant rate and constant relative distance. The concatenation we use slightly differs from how it is usually applied, as we apply it on the CTRW level, and not on each symbol of the large alphabet separately. See Section 3 for details.

1.1 Related works

RLDC and RLCC constructions. Relaxed locally decodable codes, were first introduced by [2], motivated by applications to constructing short PCPs. Their construction has a block length equal to $N = K^{1+O(1/\sqrt{q})}$. Since that work, there were no constructions with better block length, in the constant query complexity regime. Recently, [13] introduced the related notion of relaxed locally correctable codes (RLCCs), and has found applications in efficient interactive protocols, in particular, in interactive oracle proofs [21].

The work of [13] showed a construction of a q -query RLCCs with block length $N = \text{poly}(K)$. Then, [3] constructed relaxed locally correctable codes with block length matching that of [2] (up to a multiplicative constant factor that only depends on q). The construction of [3] had two main components, that we also use in the current work.

Consistency test using random walk (CTRW): Informally, given a word w , and a coordinate i we wish to correct, CTRW samples a sequence of constraints $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_t$ on w , such that the domains of \mathcal{C}_i and \mathcal{C}_{i+1} intersect, with the guarantee that if w is close

to some codeword $c^* \in C$, but $w_i \neq c_i^*$, then with high probability w will be far from satisfying at least one of the constraints. In other words, CTRW performs a random walk on the constraints graph and checks if w is consistent with c^* in the i 'th coordinate. We introduce this notion in detail in Section 2.1.

Correctable canonical PCPPs (ccPCPP): These are PCPP systems for some specified language L satisfying the following properties:

- (i) for each $w \in L$ there is a unique proof $\pi(w)$ that satisfies the verifier with probability 1,
- (ii) the verifier accepts with high probability only pairs (x, π) that are close to some $(w, \pi(w))$ for some $w \in L$, i.e., only the pairs where x is close to some $w \in L$, and π is close to $\pi(w)$, and
- (iii) the set $\{w \circ \pi_w : w \in L\}$ is an RLCC.

Canonical proofs of proximity have been introduced in [11] and have been studied in [7, 13, 20].

Lower bounds. For lower bounds, the only bound we are aware of is that of [12], who proved that any q -query relaxed locally decodable code must have a block length $N \geq K^{1+\Omega(\frac{1}{q^2})}$.

For the strict notion of locally decodable codes, it is known by [14, 23] that for $q \geq 3$ any q -query LDC must have block length $N \geq \Omega(K^{1+\frac{1}{\lceil q/2-1 \rceil}} / \log(K))$. For $q = 3$ a slightly stronger bound of $N \geq \Omega(K^2 / \log \log(K))$ is known for *linear* LDCs [23]. For $q = 2$ [17] proved an exponential lower bound of $N \geq \exp(\Omega(K))$. See also [4, 10, 19, 22, 24] for more related work on lower bounds for LDCs.

2 Proof overview

In this section we informally describe our code construction. Roughly speaking, our construction consists of two parts:

The Reed-Muller encoding: Given a message $M \in \mathbb{F}^K$, its Reed-Muller encoding is the evaluation of an m -variate polynomial of degree at most d over \mathbb{F} , whose coefficients are determined by the message we wish to encode.

Proofs of proximity: The second part of the encoding consists of the concatenation of PCPPs, each claiming that a certain restriction of the first part agrees with some Reed-Muller codeword.

Specifically, given a message $M \in \mathbb{F}^K$, we first encode it using the Reed-Muller encoding $\text{RM}_{\mathbb{F}}(m, d)$, where m roughly corresponds to the query complexity of our RLDC, and the field is large enough so that the distance of the Reed-Muller code, which is equal to $1 - \frac{d}{|\mathbb{F}|}$, is some constant, say $3/4$. That is, the first part of the encoding corresponds to an evaluation of some polynomial $f: \mathbb{F}^m \rightarrow \mathbb{F}$ of degree at most d . The second part of the encoding consists of a sequence of PCPPs claiming that the restrictions of the Reed-Muller part to some carefully chosen planes in \mathbb{F}^m are evaluations of some low-degree polynomial.

The planes we choose are of the form $\mathcal{P}_{\vec{a}, \vec{h}, \vec{h}'} = \{\vec{a} + t \cdot \vec{h} + s \cdot \vec{h}' : t, s \in \mathbb{F}\}$, where $\vec{a} \in \mathbb{F}^m$, and $\vec{h}, \vec{h}' \in \mathbb{H}^m$ for some \mathbb{H} subfield of \mathbb{F} . We will call such planes \mathbb{H} -planes. In order to obtain the RLDC with the desired parameters, we choose the field \mathbb{H} so that \mathbb{F} is the extension of \mathbb{H} of degree $[\mathbb{F} : \mathbb{H}] = m$. It will be convenient to think of \mathbb{H} as a field and think of \mathbb{F} as a vector space of \mathbb{H} of dimension m (augmented with the multiplicative structure on \mathbb{F}). Indeed, the saving in the block length of the RLDC we obtain crucially relies on the fact that we ask for PCPPs for only a small collection of planes, and not all planes in \mathbb{F}^m . The actual constraints required to be certified by the PCPPs are slightly more complicated, and we describe them next.

The constraints of the first type correspond to \mathbb{H} -planes \mathcal{P} and points $\vec{x} \in \mathcal{P}$. For each such pair (\mathcal{P}, \vec{x}) the code will contain a PCPP certifying that

- (i) the restriction of the Reed-Muller part to \mathcal{P} is close to an evaluation of some polynomial of total degree at most d ,
- (ii) and furthermore, this polynomial agrees with the value of the Reed-Muller part on \vec{x} .

In order to define it formally, we introduce the following notation.

► **Notation 3.** Let \mathbb{F} be a finite field of size n . Fix $f: \mathbb{F}^m \rightarrow \mathbb{F}$, a plane \mathcal{P} in \mathbb{F}^m , and a point $\vec{x} \in \mathcal{P}$. Denote $f_{|\mathcal{P}}^{(\vec{x})} = f_{|\mathcal{P}} \circ (f(\vec{x}))^{n^2}$. That is, the length of $f_{|\mathcal{P}}^{(\vec{x})}$ is $2 \cdot n^2$, and it consists of $f_{|\mathcal{P}}$ concatenated with n^2 repetitions of $f(\vec{x})$.

Given the notation above, if f is the first part of the codeword, corresponding to the Reed-Muller encoding of the message, then the PCPP for the pair (\mathcal{P}, \vec{x}) is expected to be the proof of proximity claiming that $f_{|\mathcal{P}}^{(\vec{x})}$ is close to the language

$$\text{RM}_{|\mathcal{P}}^{(\vec{x})} = \{Q \circ (Q(\vec{x}))^{(n^2)} : Q \text{ is the evaluation of a degree-}d \text{ polynomial on } \mathcal{P}\} \subseteq \mathbb{F}^{2n^2} . \quad (1)$$

Note that by repeating the symbol $Q(\vec{x})$ for n^2 times, the definition indeed puts weight $1/2$ on the constraint that the input $f_{|\mathcal{P}}$ is close to some low-degree polynomial Q , and puts weight $1/2$ on the constraint $f(\vec{x}) = Q(\vec{x})$. In particular, if $f_{|\mathcal{P}}$ is δ -close to some bivariate low degree polynomial Q for some small $\delta > 0$, but $f(\vec{x}) \neq Q(\vec{x})$, then $f_{|\mathcal{P}}^{(\vec{x})}$ is at least $(1 - \frac{d}{|\mathbb{F}|} - \delta)/2$ -far from any bivariate low degree polynomial on \mathcal{P} .

The constraints of second type correspond to \mathbb{H} -planes \mathcal{P} and lines $\ell \subseteq \mathcal{P}$. For each such pair (\mathcal{P}, ℓ) the code will contain a PCPP certifying that

- (i) the restriction of the Reed-Muller part to \mathcal{P} is close to an evaluation of some polynomial of total degree at most d ,
- (ii) and furthermore, the restriction of this polynomial to ℓ is close to $f_{|\ell}$.

(In particular, this implies that $f_{|\ell}$ is close to some low-degree polynomial.)

Next, we introduce the notation analogous to Notation 3 replacing the points with lines.

► **Notation 4.** Let \mathbb{F} be a finite field of size n . Fix $f: \mathbb{F}^m \rightarrow \mathbb{F}$, a plane \mathcal{P} in \mathbb{F}^m , and a line $\ell \subseteq \mathcal{P}$. Denote by $f_{|\mathcal{P}}^{(\ell)} = f_{|\mathcal{P}} \circ (f_{|\ell})^n$. That is, the length of $f_{|\mathcal{P}}^{(\ell)}$ is $2 \cdot n^2$, and it consists of $f_{|\mathcal{P}}$ concatenated with n repetitions of $f_{|\ell}$.

If f is the Reed-Muller part of the codeword, corresponding to the Reed-Muller encoding of the message, then the PCPP for the pair (\mathcal{P}, ℓ) is expected to be the proof of proximity claiming that $f_{|\mathcal{P}}^{(\ell)}$ is close to the language

$$\text{RM}_{|\mathcal{P}}^{(\ell)} = \{Q \circ (Q_{|\ell})^n : Q \text{ is the evaluation of some degree-}d \text{ polynomial on } \mathcal{P}\} \subseteq \mathbb{F}^{2n^2} . \quad (2)$$

Again, similarly to the first part, repeating the evaluation of $Q_{|\ell}$ for n times puts weight $1/2$ on the constraint that the input $f_{|\mathcal{P}}$ is a close to some low-degree polynomial Q , and puts weight $1/2$ of the constraint $f_{|\ell}$ is close to $Q_{|\ell}$.

With the proofs specified above, we now sketch the local correcting algorithm for the code. Below we only focus on correcting symbols from the Reed-Muller part. Correcting the symbols from the PCPP part follows a rather straightforward adaptation of the techniques from [3], and we omit them from the overview.

Given a word $w \in \mathbb{F}^N$ and an index $i \in [N]$ of w corresponding to the Reed-Muller part of the codeword, let $f: \mathbb{F}^m \rightarrow \mathbb{F}$ be the Reed-Muller part of w , and let $\vec{x} \in \mathbb{F}^m$ be the input to f corresponding to the index i . The local decoder works in two steps.

Consistency test using random walk: In the first step the correcting algorithm invokes a procedure we call *consistency test using a random walk (CTRW)* for the Reed-Muller code. This step creates a sequence of \mathbb{H} -planes of length $(m + 1)$, where each plane defines a constraint checking that the restriction of w to the plane is low-degree. Hence, we get $m + 1$ constraints, each depending on n^2 symbols.

Composition using proofs of proximity: Then, instead of reading the entire plane for each constraint, we use the PCPPs from the second part of the codeword to reduce the arity of each constraint to $O(1)$, thus reducing the total query complexity of the correcting algorithm to $q = O(m)$. That is, for each constraint we invoke the corresponding PCPP verifier to check that the restrictions of f to each of these planes is (close to) a low-degree polynomial. If at least one of the verifiers rejects, then the word f must be corrupt, and hence the correcting algorithm returns \perp . Otherwise, if all the PCPP verifiers accept, the correcting algorithm returns $f(\vec{x})$.

In particular, if f is a correct Reed-Muller encoding, then the algorithm will always return $f(\vec{x})$, and the main part of the analysis is to show that if f is close to some $Q^* \in \text{RM}_{\mathbb{F}}(m, d)$, but $f(\vec{x}) \neq Q^*(\vec{x})$, then the correcting algorithm catches an inconsistency, and returns \perp with some constant probability.

The key step in the analysis says that if f is close to some codeword $Q^* \in \text{RM}$ but $f(\vec{x}) \neq Q^*(\vec{x})$, then with high probability f will be far from a low degree polynomial on at least one of these planes, where “far” corresponds to the notion of distances defined by the languages $\text{RM}_{|\mathcal{P}}^{(\vec{x})}$ and $\text{RM}_{|\mathcal{P}}^{(\ell)}$. In particular, if on one of the planes f is far from the corresponding language, then the PCPP verifier will catch this with constant probability, thus causing the correcting algorithm to return \perp . We discuss this part in detail below.

It is important to emphasize that the main focus of this work is constructing a correcting algorithm for the Reed-Muller part. Using the techniques developed in [3], it is rather straightforward to design the algorithm for correcting symbols from the PCPPs part of the code. See the full version for details.

2.1 CTRW on Reed-Muller codes

Below we define the notion of *consistency test using random walk (CTRW)* for the Reed-Muller code. This notion is a slight modification of the notion originally defined in [3] for general codes. In this paper we define it only for the Reed-Muller code. Given a word $f: \mathbb{F}^m \rightarrow \mathbb{F}$ and some $\vec{x} \in \mathbb{F}^m$, the goal of the test is to make sure that $f(\vec{x})$ is consistent with the codeword of Reed-Muller code closest to f . [3] describe a CTRW for the tensor power $C^{\otimes m}$ of an arbitrary codes C with good distance (e.g., Reed-Solomon). The CTRW they describe works by starting from the point we wish to correct, and choosing an axis-parallel line ℓ_1 containing the starting point. The test continues by choosing a sequence of random axis-parallel lines $\ell_2, \ell_3, \dots, \ell_t$, such that each ℓ_i intersects the previous one, ℓ_{i-1} , until reaching a uniformly random coordinate of the tensor code. That is, the length of the sequence t denotes the *mixing time of the corresponding random walk*. The predicates are defined in the natural way; namely, the test expects to see a codeword of C on each line it reads.

In this work, we present a CTRW for the Reed-Muller code, which is a variant of the CTRW described above. The main differences compared to the description above are that

- (i) the test chooses a sequence of planes $\mathcal{P}_1, \mathcal{P}_3, \dots, \mathcal{P}_t$ (and not lines),
- (ii) and every two planes intersect on a line (and not on a point).

Roughly speaking, the algorithm works as follows.

1. Given a point $\vec{x} \in \mathbb{F}^m$ the test picks a uniformly random \mathbb{H} -plane \mathcal{P}_0 containing \vec{x} .
2. Given \mathcal{P}_0 , the test chooses a random line $\ell_1 \subseteq \mathcal{P}_0$, and then chooses another random \mathbb{H} -plane $\mathcal{P}_1 \subseteq \mathbb{F}^m$ containing ℓ_1 .
3. Given \mathcal{P}_1 , the test chooses a random line $\ell_2 \subseteq \mathcal{P}_1$, and then chooses another random \mathbb{H} -plane $\mathcal{P}_2 \subseteq \mathbb{F}^m$ containing ℓ_2 .
4. The algorithm continues for some predefined number of iterations, choosing $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_t$. Roughly speaking, the number of iterations is equal to the mixing time of the corresponding Markov chain. More specifically, the process continues until a uniformly random point in \mathcal{P}_t is close to a uniform point in \mathbb{F}^m .
5. The constraints defined for each \mathcal{P}_i are the natural constraints; namely checking that the restriction of f to the entire plane \mathcal{P}_i is a polynomial of degree at most d .

One of the important parameters, directly affecting the query complexity of our construction is the mixing time of the random walk. Indeed, as explained above, the query complexity of our RLDC is proportional to the mixing time of the random walk. We prove that if $[\mathbb{F} : \mathbb{H}] = m$, then the mixing time is upper bounded by m . In order to prove this we use the following claim, saying that if \mathbb{F} is the field extension of \mathbb{H} of degree m , and $\vec{h}_1, \dots, \vec{h}_m \in \mathbb{H}^m$ and $t_1, \dots, t_m \in \mathbb{F}$ are sampled uniformly, independently from each other, then $\sum_{i=1}^m t_i \cdot \vec{h}_i$ is close to a uniformly random point in \mathbb{F}^m . See the full version for the exact statement.

As explained above, the key step of the analysis is to prove that if f is close to some codeword $Q^* \in \text{RM}$ but $f(\vec{x}) \neq Q^*(\vec{x})$, then with high probability at least one of the predicates defined will be violated. Specifically, we prove that with high probability the violation will be in the following strong sense.

► **Theorem 5** (informal, see the full version [1]). *If f is close to some codeword $Q^* \in \text{RM}$ but $f(\vec{x}) \neq Q^*(\vec{x})$, then with high probability*

1. *either $f|_{\mathcal{P}_0}^{(\vec{x})}$ is $\Omega(1)$ -far from $\text{RM}|_{\mathcal{P}_0}^{(\vec{x})}$,*
2. *or $f|_{\mathcal{P}_i}^{(\ell_i)}$ is $\Omega(1)$ -far from $\text{RM}|_{\mathcal{P}_i}^{(\ell_i)}$ for some $i \in [m]$.*

Indeed, this strong notion of violation allows us to use the proofs of proximity in order to reduce the query complexity to $O(1)$ queries for each $i \in [m]$. We discuss proofs of proximity next.

2.2 PCPs of proximity and composition

The second building block we use in this work is the notion of *probabilistic checkable proofs of proximity (PCPPs)*. PCPPs were first introduced in [2] and [8]. Informally speaking, a PCPP verifier for a language L , gets an oracle access to an input x and a proof π claiming that x is close to some element of L . The verifier queries x and π in some small number of (random) locations, and decides whether to accept or reject. The completeness and soundness properties of a PCPP are as follows.

Completeness: If $x \in L$, then there exists a proof causing the verifier accept with probability 1.

Soundness: If x is far from L , then no proof can make the verifier accept with probability more than $1/2$.

In fact, we will use the slightly stronger notion of *canonical PCPP (cPCPP) systems*. These are PCPP systems satisfying the following completeness and soundness properties. For completeness, we demand that for each w in the language there is a unique *canonical* proof

$\pi(w)$ that causes the verifier to accept with probability 1. For soundness, the demand is that the only pairs (x, π) that are accepted by the verifier with high probability are those where x is close to some $w \in L$ and π is close to $\pi(w)$. Such proof system have been studied in [7, 20], who proved that such proof systems exist for every language in \mathcal{P} .

Furthermore, for our purposes we will demand a stronger notion of correctable canonical PCPP systems (ccPCPP). These are canonical PCPP systems where the set $\{w \circ \pi^*(w) : w \in L\}$ is a q -query RLCC for some parameter q , with $\pi^*(w)$ denoting the canonical proof for $w \in L$. It was shown in [3] how to construct ccPCPP by combining a cPCPP system with *any* systematic RLCC. Informally speaking, for every $w \in L$, and its canonical proof $\pi(w)$, we define $\pi^*(w)$ by encoding $w \circ \pi(w)$ using a systematic RLCC. The verifier for the new proof system is defined in a straightforward manner. See [3] for details.

The PCPPs we use throughout this work, are the proofs of two types, certifying that

1. $f|_{\mathcal{P}}^{(\vec{x})}$ is close to $\text{RM}_{|\mathcal{P}}^{(\vec{x})}$ for some plane \mathcal{P} and some $\vec{x} \in \mathcal{P}$, and
2. $f|_{\mathcal{P}}^{(\ell)}$ is close to $\text{RM}_{|\mathcal{P}}^{(\ell)}$ for some plane \mathcal{P} and some line $\ell \subseteq \mathcal{P}$.

Indeed, it is easy to see that the first type of proofs checks that

- (i) the restriction of f to \mathcal{P} is close to an evaluation of some polynomial Q^* of total degree at most d ,
- (ii) and $f(\vec{x}) = Q^*(\vec{x})$.

Similarly, the second type proof certifies that

- (i) the restriction of f to \mathcal{P} is close to an evaluation of some polynomial Q^* of total degree at most d ,
- (ii) and $f|_{\ell}$ is close to $Q^*|_{\ell}$.

These notions of distance go together well with the guarantees we have for CTRW in Theorem 5. This allows us to *compose* CTRW with the PCPPs to obtain a correcting algorithm with query complexity $q = O(m)$. Informally speaking, the composition theorem works as follows. We first run the CTRW to obtain a collection of $m + 1$ constraints on the planes $\mathcal{P}_0, \mathcal{P}_1, \dots, \mathcal{P}_m$. By Theorem 5, we have the guarantee that with high probability either $f|_{\mathcal{P}_0}^{(\vec{x})}$ is $\Omega(1)$ -far from $\text{RM}_{|\mathcal{P}_0}^{(\vec{x})}$, or $f|_{\mathcal{P}_i}^{(\ell_i)}$ is $\Omega(1)$ -far from $\text{RM}_{|\mathcal{P}_i}^{(\ell_i)}$ for some $i \in [m]$. Then, instead of actually reading the values of f on all these planes, we run the PCPP verifier on $f|_{\mathcal{P}_0}^{(\vec{x})}$ to check that it is close to $\text{RM}_{|\mathcal{P}_0}^{(\vec{x})}$, and running the PCPP verifier on each of the $f|_{\mathcal{P}_i}^{(\ell_i)}$ to check that they are close to $\text{RM}_{|\mathcal{P}_i}^{(\ell_i)}$. Each execution of the PCPP verifier makes $O(1)$ queries to f and to the proof, and thus the total query complexity will be indeed $O(m)$. As for soundness, if $f|_{\mathcal{P}_0}^{(\vec{x})}$ is $\Omega(1)$ -far from $\text{RM}_{|\mathcal{P}_0}^{(\vec{x})}$, or $f|_{\mathcal{P}_i}^{(\ell_i)}$ is $\Omega(1)$ -far from $\text{RM}_{|\mathcal{P}_i}^{(\ell_i)}$ for some $i \in [m]$, then the corresponding verifier will notice an inconsistency with constant probability, causing the decoder to output \perp .

A detailed discussion of proofs of proximity and the composition is available in the full version of this work [1].

2.3 Putting it all together

Below we discuss the block length and query complexity of our code.

Query complexity: For the query complexity, the mixing time of the CTRW is upper bounded by m , and for each step of the random walk, we read $O(1)$ queries from the PCPP part. Therefore, the total query complexity is bounded by $O(m)$.

Block length: As for the block length of the code, the encoding takes a message of length K and encodes it first using the Reed-Muller code of degree d over the field \mathbb{F} of size $|\mathbb{F}| = n = O(d)$, where $O(\cdot)$ depends on m , so that $K = \binom{m+d}{m} > \left(\frac{d}{m}\right)^m$. In particular, the length of the Reed-Muller part is $n^m \leq O(K)$, where, again, $O(\cdot)$ hides a constant that depends on m .

The total number of predicates (of both types) defined for the CTRW is upper bounded by $B \leq 2n^m \cdot |\mathbb{H}|^{2m} \cdot n^2 = 2n^{m+4}$, as $[\mathbb{F} : \mathbb{H}] = m$, and hence $|\mathbb{H}| = n^{1/m}$. For each such predicate, we have a PCPP part of length $\text{poly}(|\mathbb{F}|) = \text{poly}(n)$.

Therefore, the total length of our code N is upper bounded by

$$N = n^m + B \cdot \text{poly}(n) = n^{m+O(1)} .$$

A straightforward computation reveals that this is upper bounded by $C_m \cdot K^{1+O(1/m)}$, where C_m is a constant that depends on m (but is independent of all other parameters). See the full version for the exact computation.

2.4 Comparison to the previous work

Despite the high level similarity of our work with [3], we contribute several new and crucial ideas required in order to improve the parameters of RLCCs. To demonstrate the contribution of this work, we recall the previous best-known construction of RLCCs, due to [3]. The construction in [3] consists of two parts. First, they take a message M and encode it using tensor power of Reed–Solomon code which we denote by $C^{\otimes m}$. Then, for each axis-parallel line in $C^{\otimes m}$, they append a PCPP proof asserting that restriction of the codeword to the corresponding line is (close to) a Reed–Solomon code C . The relaxed local decoding procedure for a word f and a point $\vec{x} \in C^{\otimes m}$ works by running a CTRW as follows. First, the algorithm starts by choosing an axis-parallel line ℓ_1 that passes through \vec{x} , and invokes a PCPP verifier on the proof for ℓ_1 to check that $f|_{\ell_1}$ is (close to) a Reed–Solomon codeword. The algorithm continues by choosing another line ℓ_2 that intersects the ℓ_1 , and checks that $f|_{\ell_2}$ is (close to) a Reed–Solomon codeword. The procedure is repeated m times by sampling $\ell_3, \ell_4, \dots, \ell_m$, where a ℓ_{i+1} intersect ℓ_i on a uniformly random point on the line. The length of the walk m is chosen so that a uniformly random point in ℓ_m is a uniform point in the tensor code. In particular, with high probability the line ℓ_m is close to the closest global codeword.

Informally, the main idea of the analysis boils down to the following. Suppose that the line ℓ_1 is far from the closest global codeword $Q^* \in C^{\otimes m}$. That is, either (1) $f|_{\ell_1}$ is far from the Reed–Solomon code or (2) it is close to some Reed–Solomon codeword but is inconsistent with the closest global codeword $Q^*|_{\ell_1}$. In particular, using the fact that Reed–Solomon code has good distance, it follows that $\text{dist}(f|_{\ell_1}, Q^*|_{\ell_1}) = \Omega(1)$. Since after m random steps we reach a line ℓ_m that is close to the closest global codeword, i.e., $\text{dist}(f|_{\ell_m}, Q^*|_{\ell_m}) = o(1)$, then it must be the case that one of the lines $\ell_1, \ell_2, \dots, \ell_{m-1}$ is far from a Reed–Solomon, and thus the PCPP verifier rejects with high probability.

The main barrier in improving the parameters of [3] comes from the fact that ℓ_{i+1} and ℓ_i only intersect in a single point, and since we only check that the restrictions to ℓ_i 's are $\Theta(1)$ -close to the Reed–Solomon code, each random step chooses a non-corrupted point with some constant probability, which causes the algorithm to fail.

A natural idea to overcome this difficulty is to replace the lines $\ell_1, \ell_2, \dots, \ell_m$ in [3] with the planes $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_m$, such that two consecutive planes intersect on a line. Indeed, using the fact that a random line is a good sampler it will follow that corruption in \mathcal{P}_1 will propagate with high probability to \mathcal{P}_m .

The key difficulty in implementing this idea is deciding on the collection of the planes for which we require the PCPPs. For example, if we ask for a PCPP for *all* planes in \mathbb{F}^m , then the block length becomes at least \mathbb{F}^{4m} , which is a lot more than we can afford. On the other hand, if we ask for a PCPP for all *axis-parallel* planes, then we will not be able to sample a uniformly random line in each step of the random walk, as a uniformly random line will not be axis-parallel with high probability.

18:10 Relaxed Locally Correctable Codes with Improved Parameters

We overcome this difficulty by requiring PCPPs for all \mathbb{H} -planes, where \mathbb{H} is a carefully chosen subfield of \mathbb{F} , and an \mathbb{H} -plane is a plane of the form $\mathcal{P}_{\vec{a}, \vec{h}, \vec{h}'} = \{\vec{a} + t \cdot \vec{h} + s \cdot \vec{h}' : t, s \in \mathbb{F}\}$, where $\vec{a} \in \mathbb{F}^m$, and $\vec{h}, \vec{h}' \in \mathbb{H}^m$ for some \mathbb{H} subfield of \mathbb{F} .

Specifically, we choose the subfield \mathbb{H} to satisfy the following properties:

1. \mathbb{H} is sufficiently small, so that the total number of planes is small, which in turn reduces the block length of the code.
2. Mixing time for CTRW on the planes converges rapidly.

Note that the two requirements are conflicting, as fast mixing implies that the collection of the planes must be somewhat large. We show a subfield \mathbb{H} of \mathbb{F} satisfying both of these properties, thus finding a pseudo-random collection of planes which allows us to improve the parameters of the entire construction, thus obtaining the main goal of this paper. See the full version for details.

3 Concluding remarks and open problems

In this section, we first briefly discuss how to obtain a binary RLDC using the code concatenation technique. Then, we conclude the section by reviewing some open problem which we leave for future research.

3.1 Code concatenation for binary alphabet

In this paper we constructed an $O(q)$ -query RLDC $C: \mathbb{F}^K \rightarrow \mathbb{F}^N$ with block length $N = q^{O(q^2)} \cdot K^{1+O(1/q)}$, assuming that the field is large enough, namely, assuming that $|\mathbb{F}| \geq c_q \cdot K^{1/q}$. Using standard techniques it is possible to obtain a binary RLDC with similar parameters. This can be done by concatenating our code with an arbitrary binary code with constant rate and constant relative distance. Indeed, this transformation appears in [3, Appendix A], who showed how concatenating CTRW-based RLDC over large alphabet with a good binary code gives a binary RLDC that essentially inherits the block length and the query complexity of the RLDC over large alphabet. Below we provide the proof sketch, explaining how the concatenation works.

Proof sketch. Suppose that we want to construct a short binary RLCC. Let $C_{RLCC}: \mathbb{F}^K \rightarrow \mathbb{F}^N$ be the RLCC over some field \mathbb{F} with the desired block length, and let $C_{bin}: \{0, 1\}^{K'} \rightarrow \{0, 1\}^{N'}$ be an error-correcting code with constant rate and constant distance. We also assume that field \mathbb{F} is chosen so that $|\mathbb{F}| = 2^{K'}$. (To satisfy this condition, one can simply set \mathbb{H} to be a field of characteristic 2.) This assumption will allow us to have a bijection between each symbol of \mathbb{F} and binary string of length K' .

We construct the binary concatenated code $C_{concat}: \{0, 1\}^{K \cdot K'} \rightarrow \{0, 1\}^{N \cdot N'}$ as follows. Given a message $M \in \{0, 1\}^{K \cdot K'}$, we first convert it to an string in $M' \in \mathbb{F}^K$ in the natural way. Then, we encode M' using C_{RLCC} to obtain a codeword $c^* \in C_{RLCC}$. Finally, we encode each symbol of c^* using C_{bin} to get the final codeword $c \in \{0, 1\}^{N \cdot N'}$.

To prove that the concatenated code is an RLCC, Chiesa, Gur, and Shinkar proved in [3, Theorem A.4] that if C_{RLCC} admits an r -steps CTRW with some soundness guarantees, then C_{concat} admits an r -steps CTRW with related soundness guarantees. The CTRW on the concatenated code C_{concat} emulates the CTRW on C_{RLCC} by sampling planes for the CTRW on the Reed-Muller code, and instead of reading the symbols from \mathbb{F} , it reads the binary encodings of all symbols belonging to these planes.

Indeed, it is not difficult to see that if C_{RLCC} admits an r -steps CTRW with some soundness guarantees, then so does the concatenated code. We omit the details, and refer the interested reader to Appendix A in [3]. ◀

3.2 Open problems

We conclude the paper with several open problems we leave for future research.

1. The most fundamental open problem regarding RLDCs/RLCCs is to understand the optimal trade-off between the query complexity of LDCs and their block length in the constant query regime. It is plausible that the lower bound of [12] can be improved to $K^{1+\Omega(1/q)}$, although we do not have any evidence for this.
2. As discussed in the introduction, [2] asked whether it is possible to prove a separation between LDCs and RLDCs. Understanding the trade-off between the query complexity and the block length is one possible way to show such separation.
3. Another interesting open problem is to construct an RLDC/RLCC with constant rate and small query complexity. In particular, it is plausible that there exist $\text{polylog}(N)$ -query RLDCs with $N = O(K)$. It should be noted that [13] constructed constant-rate RLCCs (in fact with a rate approaching 1) and query complexity $\log(N)^{O(\log \log(N))}$.
4. Also, it would be interesting to construct RLDCs/RLCCs using high-dimensional expanders [15, 6, 5, 16]. Since there are several definitions of high-dimensional expanders, it would be interesting to state the sufficient properties of high-dimensional expanders required for RLDCs. We believe this approach can be useful in constructing constant rate RLDCs with small query complexity.

References

- 1 Vahid R. Asadi and Igor Shinkar. Relaxed locally correctable codes with improved parameters. *Electron. Colloquium Comput. Complex.*, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/142>.
- 2 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.
- 3 Alessandro Chiesa, Tom Gur, and Igor Shinkar. Relaxed locally correctable codes with nearly-linear block length and constant query complexity. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1395–1411, 2020.
- 4 A. Deshpande, R. Jain, T. Kavitha, S. V. Lokam, and J. Radhakrishnan. Better lower bounds for locally decodable codes. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pages 184–193, 2002.
- 5 Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean Function Analysis on High-Dimensional Expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 38:1–38:20, 2018.
- 6 I. Dinur and T. Kaufman. High dimensional expanders imply agreement expanders. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–985, 2017.
- 7 Irit Dinur, Oded Goldreich, and Tom Gur. Every set in P is strongly testable under a suitable encoding. *Electron. Colloquium Comput. Complex.*, 2018. URL: <https://eccc.weizmann.ac.il/report/2018/050>.
- 8 Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2004, pages 155–164, 2004.

18:12 Relaxed Locally Correctable Codes with Improved Parameters

- 9 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012.
- 10 O. Goldreich, H. Karloff, L. J. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pages 175–183, 2002.
- 11 Oded Goldreich and Madhu Sudan. Locally testable codes and pcps of almost-linear length. *J. ACM*, 53(4):558–655, 2006.
- 12 Tom Gur and Oded Lachish. A lower bound for relaxed locally decodable codes. In *31st ACM-SIAM Symposium on Discrete Algorithms*, SODA 2020, 2020.
- 13 Tom Gur, Govind Ramnarayan, and Ron D. Rothblum. Relaxed locally correctable codes. In *9th Innovations in Theoretical Computer Science Conference*, ITCS '18, pages 27:1–27:11, 2018.
- 14 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 80–86, 2000.
- 15 Tali Kaufman and David Mass. High Dimensional Random Walks and Colorful Expansion. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 4:1–4:27, 2017.
- 16 Tali Kaufman and Izhar Oppenheim. High Order Random Walks: Beyond Spectral Gap. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 47:1–47:17, 2018.
- 17 Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *Journal of Computer and System Sciences*, pages 106–115, 2003.
- 18 Swastik Kopparty and Shubhangi Saraf. Local testing and decoding of high-rate error-correcting codes. *Electron. Colloquium Comput. Complex.*, 2017. URL: <https://eccc.weizmann.ac.il/report/2017/126>.
- 19 Kenji Obata. Optimal lower bounds for 2-query locally decodable linear codes. In *Randomization and Approximation Techniques in Computer Science*, pages 39–50, 2002.
- 20 Orr Paradise. Smooth and strong pcps. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*, ITCS 2020, 2020.
- 21 Noga Ron-Zewi and Ron Rothblum. Local proofs approaching the witness length. *Electron. Colloquium Comput. Complex.*, 2019. URL: <https://eccc.weizmann.ac.il/report/2019/127>.
- 22 Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *Proceedings of the 32nd International Conference on Automata, Languages and Programming*, ICALP'05, pages 1424–1436, 2005.
- 23 David Woodruff. New lower bounds for general locally decodable codes. *Electron. Colloquium Comput. Complex.*, 2007. URL: <https://eccc.weizmann.ac.il/report/2007/006/>.
- 24 David P. Woodruff. A quadratic lower bound for three-query linear locally decodable codes over any field. In *Proceedings of the 14th International Workshop on Randomized Techniques in Computation*, RANDOM 10, pages 766–779, 2010.
- 25 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1):1:1–1:16, 2008.
- 26 Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012.